AFRL-IF-RS-TR-2004-29
Final Technical Report
February 2004

ARCHITECTURE AND PROTOTYPE OF AN AMBIENT COMPUTATIONAL ENVIRONMENT

University of Kansas Center for Research, Incorporated

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J960

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

AFRL-IF-RS-TR-2004-29 has been reviewed and is approved for publication.

APPROVED: /s/

ROBERT L. KAMINSKI
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** FEBRUARY 2004	**3. REPORT TYPE AND DATES COVERED** Final Jun 00 – Jun 03

4. TITLE AND SUBTITLE ARCHITECTURE AND PROTOTYPE OF AN AMBIENT COMPUTATIONAL ENVIRONMENT	**5. FUNDING NUMBERS** C - F30602-00-2-0581 PE - 62301E PR - J960 TA - 21 WU - A1
6. AUTHOR(S) Gary J. Minden, Joseph B. Evans, Arvin Agah, Jeremiah W. James, and Leon Searl	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Kansas Center for Research, Incorporated 2335 Irvin Hill Road Lawrence Kansas 66044-7612	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency AFRL/IFG 3701 North Fairfax Drive 525 Brooks Road Arlington Virginia 22203-1714 Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2004-29

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Robert L. Kaminiski/IFG/(315) 330-1865/ Robert.Kaminiski@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 Words)*

This Final Report describes the performer's effort of the design, development and initial prototype implementation of an Ambient Computational Environment (ACE). The concept began with the idea that computation resources, in the broadest sense, are readily available in our offices, conference rooms, auditoriums, and hallways. Users co-opt, with authorization, the computational resources within their proximate area. Users access computational services that are long-lived and extremely robust.

14. SUBJECT TERMS Computational Environment, Wireless Networking, Video Conferencing	**15. NUMBER OF PAGES** 14
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

List of Tables

Architecture and Prototype of an Ambient Computational Environment

1 Architecture and Prototype of an Ambient Computational Environment

We proposed to research the design, development and initial implementation of an Ambient Computational Environment (ACE). The concept begins with the idea that computation resources, in the broadest sense, are readily available in our offices, conference rooms, auditoriums, and hallways. Second, users co-opt, with authorization, the computational resources within their approximate area. Third, users access computational services that are long-lived and extremely robust. And fourth, users interact in multiple ways with the Ambient Computational Environments

Second, we proposed to research the impact of ACEs on high-performance networking systems. The type of traffic in an ACE is likely to be quite different from conventional ftp, web transfers, and large dataset access. Further, we proposed to investigate the mechanisms needed to secure transmission of ACE content over widely distributed next generation internets.

ACEs represent a significant application and networking protocols driving Next Generation Internets. Networking Systems research has had a difficult time answering the question, "Where's the data?" Certainly, high-resolution imagery and video is one source. We believe that low-latency, highly reliable, interactive traffic, as typified by ACEs, is a major driver for emerging network protocols. We proposed to develop such traffic applications, the necessary network protocols, and measure network performance, and do so in the wide-area networking context.

While a comprehensive research program was proposed and a project initiated in June 2000; the project was de-scoped by the government in January 2002 and terminated in June 2002. During the limited active time of the project we were able to specify and design many of the software components for ACE and started some of the software development. This report provides an overview of the project, describes the documentation presented in a series of technical reports and project management.

1.1 Ambient Computational Environments

The following vignettes illustrate our concepts. Herman completes editing his presentation in his office environment. He picks up a small lightweight device, we call a Personal Interactive Device (PID), and heads down the hall toward the conference room. The conference room is equipped with tabletop display screens, video/computer display projectors, sound system, microphone system, controllable video camera, and controllable lighting. Once there, Herman identifies himself through his organizer and requests his conventional working context be brought up on one of the table-top displays. He requests, and is granted, access to the conference room resources. From his working context, he arranges to bring his presentation to one of the data projectors, sets lighting levels , arranging video cameras pointed toward the speaker position, directing a remote video feed to a second projector, and so forth as he sets up for his presentation. Herman has co-opted the conference room's resources to support his specific presentation. He knows he can do the same thing in any of the center's conference rooms or colleague's offices.

The last time Holly checked, her working context had been running for 497 days. Today, MIS is scheduled to swap her primary computer, display, and input station for a newer model. When the MIS support person arrives with the new equipment, Holly clears a path to her "computer", he quickly unplugs the display, keyboard, and computer modules to make room for the new equipment, disregarding the fact that the computer is running and, for the time being, is Holly's port to the computational utility. He plugs in the new equipment, turns it on, and leaves. Holly identifys herself to the new machine with her organizer and it quickly displays her working context. Tomorrow when she checks the "uptime", it will show 498 days.

A few weeks later, Herman has a second presentation. Only this time, when he enters, the conference room senses his entry and turns on the ambient light. Herman sits at a table top station and presses his thumb against a reader. His thumb print identifies him and brings up his working context on the nearby display. Herman directs, through gestures, voice commands, and conventional computer commands, to put the presentation on the right screen, point "that video camera" at "that seat", "put the remote video feed on the left screen", and so forth. The conference room reacts to Herman's voice commands, gestures, and computer mediated commands. Herman feels immersed in an Ambient Computational Environment.

The concepts of Ambient Computational Environments are the following:

Computational resources are readily available throughout the space in which people move. By "computational resources " we mean CPU cycles, memory, storage, display, wired and wireless communications, sound input and output, video input and output, i.e. anything connected with computing.

Users co-opt computational resources in their vicinity for their use.

Computational sessions are long-lived, and mobile beyond the extent of individual machines or instantiations.

The computational environment re-acts to user voice commands, gestures, and computer commands and maintains an individual model of how specific users act.

Our vision embeds low cost and high performance future computational units in our everyday environment. Offices will have computational resources, conference rooms will have computational resources, taxis will have computational resources, and airplanes will have computational resources, along with a multitude of other environments. Our vision distributes computation throughout our environment which can be co-opted for our use with proper authorization. Identification to the environment either permits or denies access to the local computational environment. One only need provide their valid identification to access available resources. This is contrary to the concept that we will carry around with us a "tool belt" of information appliances to address our needs. The authors of this proposal are tired are carrying ten pound computers, pagers, cell phones, and other personal devices everywhere they go.

The Ambient Computational Environment is different from conventional mobile computing or agent systems. In mobile computing, users lug their own computer to different locations and their computer maintains their working context. In other cases, users access a central location for their files and context depending on an underlying network communications infrastructure. In either case, the focus is on communications from a fixed resource, e.g. the laptop, back to a single central system. In agent systems, programs, called "agents," are launched into an interconnected set of computers to compute and carryout tasks for the launching entity. However, any movement of the user is ignored. Agents are disembodied from the actual user. Our vision is that users "carry" their working context with them either via mechanical means (a personal organizer) or individual identification and the computational environment is available and adapts to the individual's requirements and behavior.

Our vision is distinct from other recent concepts outlining future computer system organizations presented by Norman [1] and Kozyrakis and Patterson [2]. In these proposed concepts, the low cost and high performance of future computational units will be used to create ever more powerful mobile computers. Norman argues for a plethora of "information appliances." Kozyrakis and Patterson propose integrating the functions of calculators, personal organizers, multiple wireless phone services, paging, and audio/visual remote controls, and broadcast radio into a single personal unit. They further argue that multi-media applications and data streams will change the basic underlying computer architecture.

Sun's Jini™ architecture [3] considers "mechanisms for machines or programs to enter into a federation where each machine or program offers resources to other members of the federation." While Jini™ technology is one possible technology base for Ambient Computational Environments, it is only a network technology and does not describe the necessary services to build a robust environment, nor the necessary knowledge systems to provide intelligent reaction to user commands.

ACEs will have significant impact on network usage. Implementing persistent storage will require extremely low latency communications protocols. Interaction between ACE components is likely to be more of a transactional nature rather than today's client/server (GET/PUT) nature of the world wide web. Multiple high definition video and audio streams will require at least two orders of magnitude capacity over today's streaming video sessions. Understanding these shifts in network load and behavior is a major component of the proposed research.

Section 1.2 describes important research problems necessary to implement an Ambient Computational Environment.

1.2 Research Topics

We have identified seven areas of research necessary to build Ambient Computational Environments:

- System architecture of mobile programming contexts; access to contexts from multiple, remote locations; and persistent storage of contexts.
- Network protocols to support persistent storage systems over wide areas, transactional based communications, and highly mobile routing techniques and network behavior under ACE loads.
- Tools and techniques for access to and interaction with environments.
- Language and run-time systems to support long-lived, mobile computing contexts.
- Task negotiation among multiple user workspaces.
- Context-sensitive information retrieval.
- Disambiguation of spoken commands and gestures within different environments and multiple user contexts.

These research topics represent the significant range of activities necessary to build an ACE. Access to an ACE requires communication services, as well as efficient and intuitive user interfaces, which in turn require innovative interaction devices and methods. Environments that support long-term, mobile computations place new demands on the programming languages, underlying program support systems, and communication systems. Management of user intentions, history, and command conflicts, as represented by user workspaces, require negotiation mechanisms among multiple users, their access devices, the environment, and ongoing computations and services. We believe information management and retrieval services will be improved by utilizing user workspaces. Finally, to build truly responsive rooms, we need necessary techniques to understand the user's commands. A significant effort in combining multiple command sources, such as speech input, gesture recognition, tactile input, and other sensor input is necessary to clearly understand the command and respond in an appropriate manner.

This project focused on the first three research areas: System architecture, Network protocols and behavior, and access to ACEs The remaining part of this section describes these research areas in more detail and outlines a specific set of research problems raised by the Ambient Computational Environment concept.

1.2.1 ACE System Architecture

Ambient Computational Environments require significant re-thinking of how applications are developed and structured and the nature of communications among multiple components within the ACE. Today's computing/networking is characterized by client-server relationships between distinct computers and by local applications that interact with their environment through four primary portals: local graphic display, local keyboard and mouse inputs, local file systems, and network connections. One might add a fifth application portal: local temporary storage. To pick-up an application from one computer and deposit it on another and expect the application to continue operating is well beyond today's operating system and programming language constructs. Even to consider re-routing an application's graphic display output and keyboard/mouse inputs to another access point is extremely difficult.

We proposed a threefold approach to building ACE computational components:

(1) Wrapping existing applications to intercept and re-route user input/output streams (keyboard, mouse, and graphic display);

(2) A run-time environment suitable for moving access points for input/output, file system, and network connections; and

(3) Programming language constructs to make the creation of ACE components easier.

As a first step, we considered wrapping existing applications, written for either Linux or Windows NT, to intercept and re-route user input/output streams (keyboard, mouse, and graphic display). This is similar to commercial products such as Netopia's Timbuktu and experimental tools such as Cambridge Research Laboratories' VNC frame buffer replication. The advantage of this approach is that we can re-route access from multiple locations to fixed (existing) applications. This approach provided an early concept demonstration and allowed us to incorporate limited existing applications into the ACE environment.

We also designed and prototyped a runtime environment for controlling ACE devices. This work is described further in [4].

We were not able to develop the propsed programming language constructs due to the de-scoping of the contract.

1.2.2 Impact of ACEs on Network Performance, Behavior, and Infrastructure

ACEs use the network in an entirely different manner than traditional client/server/web systems. The replication of contexts to insure persistent state requires extremely low latency, extremely fast transactions. The ability to move computational contexts through the network and the need to re-assign input/output streams means that we must develop protocols and routing mechanisms that attend to individual computational sources and sinks, rather than physical hosts and application ports. Further, we anticipate ACEs to eventually cross traditional internet address spaces, a common criteria to allow/dis-allow communications. ACEs re-define how computational structures operate across networks.

We initially identified three kinds of ACE network transactions that differ significantly from today's network traffic:

(1) ACE traffic will be transactional in nature. There will be short messages that must be delivered in short order, in a reliable fashion, and crossing the wide area network.

(2) ACEs will support multiple, high definition video, audio, and graphic streams; each stream at least two orders of magnitude greater than todays limited streaming video/audio.

(3) ACEs will support highly interactive user input/output systems across the wide-area network.

We anticipate that we will need to carefully consider the network protocols necessary to carry out quick transactions, possibly across a wide area network, to maintain consistency and effect reasonable interactions with the user's contexts. As examples of transactional network traffic, we consider mouse and/or other pointer input; keyboard input; and/or voice command input.

A second, important component of the proposed work was the careful measurement and analysis of the impact of Ambient Computational Environments on the network performance and behavior. The deployment of ACEs by internet providers will have significant impact on the services which customers expect from the network, and consequently will impact the management and control of the network. For example, customers working from home require access to their work environment. Communication between customers at home and the ACE services at work requires reliability and quality of service from the network far beyond what the typical Internet user receives today. Directory services implemented using protocols such as LDAP with customer information would likely be required to support the allocation of the appropriate network resources to the customer connections.

In this vein, we obtained support from and worked with Sprint to extend the prototype ACE to multiple sites. We designed and build a demonstration prototype (with Sprint support) at Sprint and interconnected the two ACE sites with the Sprint / KU testbed network.

Detailed characterization of the impact of ACE concepts on local and wide area networks was not carried out due to the de-scoping of the contract.

1.2.3 Programming Languages and Run-time Systems

The implementation of an ambient-based programming language also presents challenges. An obvious approach is to use the Java VM since implementations of it are becoming ubiquitous. This approach would at a minimum require moving live objects from JVM to JVM, and this is not currently possible. It is also unlikely that support for moving live objects will be added any time in the near future, including JDK 2.0 . The problem is that moving live objects would require significant changes to the linker semantics of the JVM, and since the JVM's security model is intimately tied with the linker's semantics, supporting mobile live objects may require radical architecture changes.

Another approach is to design a successor to the JVM that would support an ambient-based programming language as well as support existing Java classes. This would allow existing code to be leveraged in the new system, but since the loader and linker of the new VM are likely to differ from the JVM, it would probably not be possible to support all Java programs. Classes inheriting from java lang.ClassLoader, for example, would not load. The goal, however, would be to preserve compatibility with the existing JVM as best as possible, and most classes would load unchanged.

The successor JVM would include run-time support for storage management and ambient mobility, but it would make no special provision for fault-tolerant computing. In particular, a running VM may provide persistent storage or journaling file systems in case of unexpected termination. This extra robustness is an optimization and may improve throughput, but it would not affect the underlying programming model since the model makes few assumptions about quality of service.

Other virtual machine architectures, such as Objective CAML [20] were also considered.

However, due to the de-scoping of the contract, this work was not initiated.

2 Accomplishments

During the shortened duration of the project we were able to design the major components of an Ambient Computational Environment and implement prototypes for many of the devices. In particular we implemented prototypes to control cameras, projectors, sound input and output, video input and output, fingerprint readers, iButton token readers, a services directory, and workspace access. This work is described in a set of technical reports described in Table 1 and is available from the Information and Telecommunications Technology Center.

ITTC-FY2001-23150-01	Renzo Hayashi, Leon Searl Gary Minden	The Ambient Computational Environments Architecture for Reliable, Secure, and Pervasive Computing	December 2000
ITTC-FY2002-23150-02	Renzo Hayashi, Leon Searl Gary Minden	ACE Architecture Design	December 2001
ITTC-FY2001-23150-03	Leon Searl, Gary Minden	ACE General Service Daemon Data Thread, Command Semantics and Client Command Design	January 2001
ITTC-FY2001-23150-04	Renzo Hayashi, Leon Searl Gary Minden	ACE Project Service Command Language Specifications Version 1.0	July 2000
ITTC-FY2001-23150-05	Leon Searl, Gary Minden	Ambient Computing Environment: ACE Service Interface Specification	January 2001
ITTC-FY2001-23150-06	Renzo Hayashi, Leon Searl Gary Minden	ACE Service Directory Interface Specification	May 2001
ITTC-FY2001-23150-07	James Mauro Leon Searl, Gary Minden	ACE Connection Interface Specification Version 0.9	January 2001
ITTC-FY2001-23150-08	James Mauro Leon Searl, Gary Minden	ACE Project ACE Authorization Interface Specification Version 0.1	June 2001

Table 1: List of the ACE Technical Reports

3 Project Information

The ACE project ran from June 1, 2000 through June 30, 2002. The original contract called for a June 1, 2000 through December 31, 2003 period and $1,460,991 ($1,320,000 Federal and $140,991 The University of Kansas) budget. In January 2002 the government de-scoped the contract to run through June 30, 2002 and a budget of $650,000 (Federal).

3.1 Project Personnel

Professors Gary J. Minden, Joseph B. Evans, Arvin Agah, and Jeremiah W. James directed the project. Research Engineer Leon Searl helped organize the project and direct the graduate students. Four graduate students and four undergraduate students worked on this project: Julie Johnson, Franklin Jones, Sivaprasath Murugeshan, Rajiv Ramanasankaran, Vidyaraman Sankaranarayanan, Eric Akers, Ramakrishnan Kalicut, James Mauro, Sreenivas Penumarthy, Renzo Hayashi, Prasanna Ramasubram

Condor Chou, Balaji Rajagopalan, Jedrzej Miadowicz, and Ramu Narapparaju. Many of these students continued their work on ACE after the project ended and earned their Masters of Engineering degree on this project.

3.2 Project Equipment

The ACE project was supported by a National Science Foundation Research Infrastructure grant EIA-9972843 and The University of Kansas. No equipment was purchased under the contract.

4 Conclusion

During the limited duration of the project, we successfully designed and implemented prototypes for an Ambient Computational Environment. The design is documented in a series of technical reports and students continue to work on the concept.

References

[1] Norman, D. A., The Invisible Computer, The MIT Press, Cambridge, MA, 1998.

[2] Kozyrakis. C. E, and Patterson, D. A., "A New Direction for Computer Architecture Research," IEEE Computer, Vol. 31, No. 11, November, 1998, pg. 24-32

[3] Waldo, J. "Jini™ Architecture Overview," Sun Microsystems, Palo Alto, CA, 1998.

[4] Hayashi, R., Searl, L, and Minden, G., "The Ambient Computational Environments Architecture for Reliable Secure, and Pervasive Computing," December 2000, ITTC Technical Report ITTC_FY2001-TR-23150-01.